UNIVERZITA PALACKÉHO V OLOMOUCI

PEDAGOGICKÁ FAKULTA

Katedra technické a informační výchovy

Bakalářská práce

František Peiker

Srovnání vybraných Javascriptových frameworků v kontextu tvorby jednoduchých webových aplikací

Olomouc 2022 vedoucí práce: Mgr. Jan Kubrický, Ph.D.

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně a uvedl jsem v ní veškerou literaturu a ostatní informační zdroje, které jsem použil.

V Olomouci dne 20. 4. 2022 .....

vlastnoruční podpis

Poděkování

Rád bych poděkoval svému vedoucímu bakalářské práce panu Mgr. Janu Kubrickému Ph.D. za odborné vedení, za pomoc a cenné rady, které mi pomohly tuto práci zkompletovat.

Anotace

Bakalářská práce se zabývá dvěma Javascriptovými frameworky: React a Angular. V dané problematice se práce především zaměřuje na jejich srovnání z pohledu uživatele i programátora, a který z nich je vhodnější pro tvorbu jednoduchých webových aplikací. Práce je také soustředěna na tvorbu webových stránek samotných.

Klíčová slova

JavaScript, HTML, framework, React, Angular, web

Abstract

This bachelor thesis has been observing two JavaScript frameworks: React and Angular. The thesis has mainly aimed for their comparison from the view of a user and a programmer, and which of them is more suitable for developing simple web applications. It has also been focusing on web development itself.

Key words

JavaScript, HTML, framework, React, Angular, web

Obsah

Úvod 7
Teoretická část 9
1 Základy pro tvorbu webových stránek 10
1.1 Statická webová stránka vs. dynamická webová stránka 11
1.2 Jednoduchá webová aplikace 12
2 JavaScript 14
3 Javascriptové frameworky 17
4 React 19
4.1 React elementy a JSX 20
4.2 Komponenty v Reactu 20
4.2.1 Přizpůsobení komponentů pomocí argumentů props 21
4.2.2 Stylování komponentů 21
4.2.3 Použití objektu state na třídy komponentů 21
4.2.4 Správa komponentů pomocí souboru nástrojů Hooks 22
4.2.5 Sdílení komponentů pomocí kontextu 22
4.3 Ovládání DOM a okénkových událostí 22
4.4 React forms 23
4.5 Použití metody Redux 23
5 Angular 24
5.1 Direktivy Angularu 26
5.2 Výrazy Angularu 26
5.3 Systém MVC 26
5.3.1 Model 27
5.3.2 Pohled 27
5.3.3 Controller 27
Praktická část 28
6 Srovnání různých webových aplikací jednotlivých frameworků 29
6.1 Kritéria nástroje Lighthouse 29
6.2 Testování webových aplikací nástrojem Lighthouse 31
7 Jednoduchá webová aplikace 34
7.1 Stanovení předpokladů 35
7.2 Srovnání frameworků za pomoci nástroje Lighthouse 36
7.3 Srovnání zatížení paměti jednotlivými frameworky 38
7.4 Srovnání samotných projektů 38

7.4.1 Vytvoření nového projektu 39
7.4.2 Orientace v projektu 41
7.4.3 Přehlednost při psaní kódu 43
7.4.4 Potřebný počet řádků zdrojového kódu 44
7.4.5 Kvalita dokumentace 44
7.5 Vyhodnocení srovnávání 47
Seznam použitých zdrojů 50

Úvod

V současnosti se internet rozvíjí ohromnou rychlostí a jazyky pro tvorbu webových stránek HTML[1] a CSS[2], již nezvládají držet krok s potřebami uživatelů, a tak bych řekl, že patří minulosti, i přes skutečnost, že stále slouží jako základy pro tvorbu webu. Dnešní moderní přístupy vyžadují ovládání databází a jazyka SQL[3] a jejich následné propojení s webovou stránkou pomocí jazyka PHP[4]. K tomu všemu je zapotřebí také JavaScript, který zajišťuje dynamičnost a responzivitu stránek. V dnešní době je JavaScript jedním z nejpopulárnějších a nejrozšířenějších programovacích jazyků. Popularita Javascriptu také vězí v tom, že se jako jediný nachází na téměř všech počítačích. Všechny moderní počítače s operačním systémem Windows obsahují již předinstalovaný internetový prohlížeč Microsoft Edge, ve kterém si kdokoli může otevřít vývojářské nástroje, a v konzoli bez instalace jakéhokoliv vývojového prostředí programovat. JavaScript se tedy dá používat v internetovém prohlížeči samotný, jako tzv. Vanilla JavaScript, ale jeho možnosti jsou zde omezené. Proto se dnes k Javascriptu nabízí rozšíření o knihovny a frameworky.

Ve své bakalářské práci bych se rád zaměřil na porovnání dvou Javascriptových frameworků. Vybral jsem si dva z nejpoužívanějších a nejznámějších: Angular a React. Nejprve však v teoretické části stručně popíšu historický vývoj tvorby webových stránek za pomocí jazyků HTML a CSS. Dále navážu na samotný JavaScript a proč je tak populární. V praktické části předvedu jednoduchou webovou aplikaci, která má podobu kvízu na znalosti z angličtiny, jako návaznost na můj druhý obor angličtinu. Aplikaci naprogramuji v obou uvedených frameworcích a následně je srovnám.

Téma jsem si vybral, protože JavaScript byl prvním jazykem, ve kterém jsem se učil programovat. Vybral jsem si ho na doporučení kamaráda a hned jsem si tento programovací jazyk zamiloval, protože má intuitivní syntax a především, jak jsem již zmiňoval, se v něm dá programovat jednoduše rovnou v prohlížeči a není třeba pro práci s ním žádné integrované vývojové prostředí, a to pro mě jako začátečníka programátora bylo vítané plus. Pro práci s přídavnými knihovnami nebo frameworky už je ale třeba si nějaké vývojové prostředí nainstalovat. Frameworky je dobré využívat hlavně pro vytváření rozsáhlejších a složitějších webových aplikací, jelikož nabízejí jednoduché strukturování zdrojového kódu aplikace

[1] HTML – z anglického HyperText Markup Language. Značkovací jazyk pro tvorbu webových stránek.
[2] CSS – z anglického Cascading Style Sheets. Jazyk pro popis vizuálního zpracování webových stránek.
[3] SQL – z anglického Structured Query Language. Jazyk pro tvorbu a úpravu databází.
[4] PHP – z anglického Hypertext Prepocessor. Skriptovací programovací jazyk pro tvorbu webových stránek.

do složek a souborů a jejich následné propojení. Používají je především internetové společnosti, které si zakládají na grafické úpravě a úhledné vizualizaci obsahu. Jedná se především o sociální sítě, kde je potřeba správné nastavení příspěvků a fotek, o reklamách nemluvě. V praktické části budu pro naprogramování webové aplikace používat softwarový systém Node.js a pro správu zdrojového kódu editor Visual Studio Code.

Hlavním cílem této práce je porovnat dva Javascriptové frameworky React a Angular. Budu je srovnávat na mnou vytvořené webové aplikaci, identické v obou frameworcích. Kritéria pro srovnávání budou z pohledu uživatele i programátora, a především se bude jednat o přehlednost při psaní kódu, množství nutného zápisu zdrojového kódu, dokumentaci, rychlost načítání a zatížení paměti jednotlivými frameworky.

Teoretická část

1 Základy pro tvorbu webových stránek

Obrázek 1: Logo HTML a CSS

Při používání internetu se propojují dva základní druhy počítačů: servery, které ukládají a poskytují informace a klienty, kteří tyto dokumenty přebírají a zobrazují uživatelům. Internet je od základů vytvořen jako prostředek, který není závislý na hardwaru klienta. Internetové dokumenty tak obsahují pouze text, a všechny přídavné materiály jako obrázky a jiné grafické objekty jsou uloženy v samotných souborech a odvolává se na ně pomocí odkazů. Internetový dokument na straně klienta zobrazují internetové prohlížeče jako Microsoft Internet Explorer, Google Chrome nebo Firefox. (Laurenčík, 2019)

HTML je značkovací jazyk pro tvorbu webových stránek. Jeho počátky sahají do roku 1980, kdy fyzik Tim Berners-Lee navrhl a vytvořil systém ENQUIRE pro výzkumníky z CERNU, aby si zde mohli mezi sebou sdílet dokumenty. V roce 1989 Berners-Lee navrhl první hypertextový systém, který se začal podobat internetu, jak ho známe dnes a v roce 1990 specifikoval jazyk HTML a vytvořil pro něj prohlížeč společně se serverovým softwarem. Od té doby se HTML dále rozšiřoval a zdokonaloval ve verzích HTML2, 3, 4, až do současné verze HTML5. (Berners-Lee, 1990)

Řídící příkazy jsou uložené v souboru typu HTML, které zajišťují správné zobrazení internetového dokumentu v prohlížeči. Každý HTML soubor by měl také obsahovat formátování, což je možné provést dvěma způsoby, a to buď přímím zapsáním údajů o barvě, velikosti, zarovnání písma apod. pomocí příkazů HTML, nebo zapsáním informací o formátování odděleně pomocí kaskádových stylů, a v jazyku HTML se na tyto styly odkazovat. (Laurenčík, 2019)

Kaskádovými styly je myšlena informace, která je uložená pod určitým jménem a obsahuje pravidla pro formátování webové stránky, ke které jsou kaskádové styly připojeny. Kaskádové styly se popisují v jazyce CSS a jsou velice účinné, protože od sebe umožňují rozdělit kód a formátování, a tak činí tento kód přehlednějším. Kaskádové styly vznikly

okolo roku 1997 jako soubor metod pro grafickou úpravu webových stránek. V poslední době se již obyčejné HTML formátování téměř nikde neobjevuje a používají se především kaskádové styly, protože přináší hned několik výhod jako rozsáhlejší možnosti formátování, jednodušší údržbu webové prezentace, oddělení struktury od stylu, nebo možnost dynamicky měnit kaskádové styly jednotlivých elementů s pomocí Javascriptu. (Janovský, 1999-2022)

Při věnování se tvorbě webových stránek si nelze nevšimnout, jak se jazyk JavaScript stává stále populárnějším. S ohledem na to, jak se v posledních několika letech výpočetní technika rozvíjí do všech odvětví, včetně školství, výpočetní oblast webových stránek nabývá na nevídané popularitě. Důvodů je hned několik. Internet je v dnešní době jednoduše dostupný ze spousty zařízení a díky zmenšování mobilních telefonů a jiných chytrých zařízení má k internetu přístup v podstatě kdokoli a odkudkoli. Rozvíjejí se webové prohlížeče, internetové obchody a sociální sítě, které se předhánějí v popularitě a tím stále přicházejí s modernějšími prostředky, se kterými se snaží zaujmout uživatele. V neposlední řadě je zde také jednoduchý přístup k programovacím jazykům a vývojovým prostředím, ve kterých se s nimi pracuje. (Žára, 2015)

„Základní koncepty a jazyky WWW však zůstávají a postupným vylepšováním upevňují web na předním místě každodenních životních činností. Práce s HTML stránkami, jejich design a postupné oživování pomocí skriptů jsou tak více a více považovány za základní IT vzdělání.“ (Žára, 2015, s. 11)

1.1 Statická webová stránka vs. dynamická webová stránka

Statická webová stránka je místo, které se skládá z více HTML souborů. Webový server je prostředek pro odesílání webové stránky na základě požadavků od internetového prohlížeče, které vznikají při odkázání se uživatele na webovou stránku. Obsah statické webové stránky se nemění, což znamená, že neobsahuje žádné dynamické prvky, které by se mohli v průběhu užívání stránky aktualizovat. Tento koncept se již moc nevyužívá kvůli stále se zvyšujícím potřebám uživatelů, ale i dnes lze najít statické stránky například v podobě blogů. (Informace o webových aplikacích, 2021)

V dnešní době se stále více popularizují dynamické webové stránky, kvůli jejich vizuálnímu vzhledu a jejich intuitivní responzivitě. Když webový server přijme požadavek na dynamickou stránku, předá stránku aplikačnímu serveru, který odpovídá za dokončení načtení stránky. Aplikační server přečte kód stránky a dokončí její načtení podle instrukcí ze skriptů nebo tagů. Webová aplikace je potom místo, které obsahuje dynamické stránky

neboli stránky s částečně nebo úplně neurčeným obsahem. Konečný vzhled těchto stránek určí aplikační server podle uživatelových příkazů při jejich užívání. (Informace o webových aplikacích, 2021)

Statická webová stránka	Dynamická webová stránka
Obsah stránky je neměnný a každá stránka obsahuje pouze HTML a CSS kód.	Obsah stránky je dynamický a mění se podle uživatelových příkazů.
Zobrazuje stále stejný design a obsah všem uživatelům navštěvujících stránku.	Nabízí unikátní směs dynamického obsahu a interaktivních prvků.
Aktualizace stránky mohou vypadat hrozně a výt náchylné k erorům.	Aktualizace stránky jsou mnohem jednodušší.
K vytvoření statické webové stránky je třeba pouze znalost HTML.	K vytvoření dynamické webové stránky je třeba znalostí jazyků PHP, Javascriptu, SQL, …
Jelikož stránku vytváří programátor, dává mu to více flexibility a kontroly nad ní.	Uživatel má plnou kontrolu nad aktualizacemi designu a obsahu stránky.

Tabulka 1: Tabulka srovnání vlastností statického a dynamického webu

1.2 Jednoduchá webová aplikace

Tento koncept není nikde pevně zaveden, a tudíž nikde nelze dohledat jedna konečná a oficiální definice. Jednoduchou webovou aplikací je z mého pohledu myšleno to, že je pro její vytvoření třeba pouze znalost HTML, CSS a Javascriptového frameworku a další nástroje pro tvorbu webových aplikací v ní nejsou použity. Jednoduchá aplikace také obsahuje jen několik dynamických prvků, není zamýšlena pro pozdější aktualizace, spíše jako jednorázový projekt, má poměrně nízký počet řádků ve zdrojovém kódu s použitím jednoduchých funkcí a metod, obsahově je spíše zamýšlena buď jen jako malý osobní projekt pro testování, nebo ukázkový projekt do výuky nebo do zaměstnání a má jen jednu nebo málo funkčností.

Ostatní webové aplikace, jako Facebook, Amazon nebo Seznam už jsou sofistikovanější. Opět zde není žádné pravidlo nebo definice, která by říkala, že jsou tyto aplikace středně složité nebo složité, ale už od pohledu lze vidět, že plní více účelů, například Seznam slouží jako vyhledávač, zdroj novinek a zpráv a e-mailová služba. Tyto velké/složité webové aplikace využívají více nástrojů pro tvorbu webu, jako je propojení s databází, propojování vícero webů za pomocí API[5] dokumentací apod. To všechno je také umožněno tím, že na složitějších webových aplikacích v obrovské většině pracuje více programátorů pod jednou internetovou společností, která je financuje a dodává pokyny pro případné budoucí zlepšování a aktualizaci webu.

[5] API – z anglického Application Programming Interface. Rozhraní pro programování aplikací.

2 JavaScript

Obrázek 2: Logo Javascriptu

Kromě jazyků HTML a CSS se tedy k tvorbě webových stránek využívá také JavaScript, který, jak jsem již zmiňoval, zajištuje dynamičnost a responzivitu webových stránek. V posledních několika letech se JavaScript rozšiřuje obrovskou rychlostí, protože je to mocný a všestranný programovací jazyk. (Pehlivanian, 2014)

Jeho počátky sahají do roku 1995, kdy ho poprvé představil Brendan Eich. JavaScript měl být pouhým doplňkem k jazyku Visual Basic od Microsoftu, kde měl za úkol pouhou validaci formulářů a drobnou manipulaci s obsahem stránky. Z počátku byl vyvíjen pro webový prohlížeč Netscape 2 a stal se ECMA[6]-262 standardem v roce 1997. Poté, co Netscape předal JavaScript společnosti ECMA, organizace Mozilla pokračovala ve vývoji jazyka pro prohlížeč Firefox. Postupem času vyšlo několik verzí s úpravami a přidáním modernějších prvků. Aktuálně je nejnovější verze Javascriptu ES6[7], která vyšla v roce 2015 s nejnovější aktualizací v roce 2018. (JavaScript History, 1999-2022)

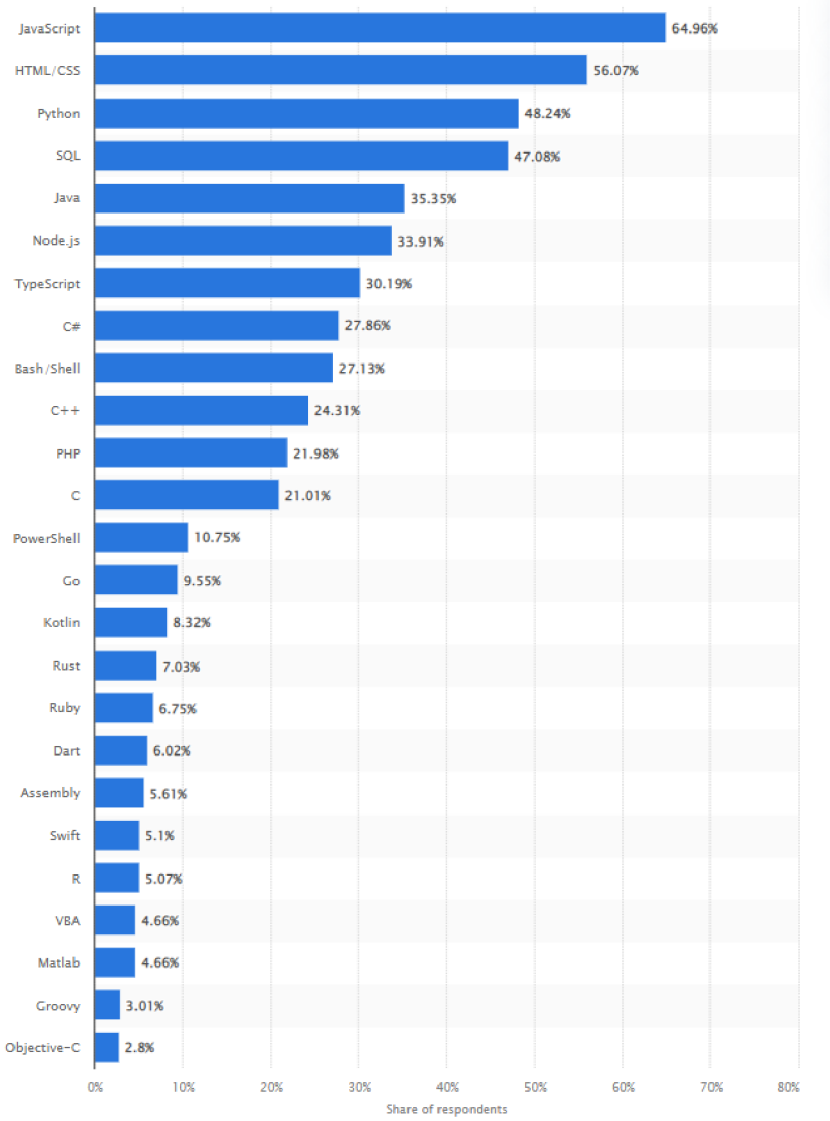
Souhrou historických událostí, JavaScript, který byl navržený hlavně pro skriptování v prostředí internetového prohlížeče, podědil po svých předchůdcích nezvyklou kombinaci vlastností. Je to dynamicky typovaný jazyk, což znamená, že se o datových typech proměnných rozhoduje až při spuštění programu, jeho syntax vychází z jazyka C, některá rozhraní, jako proměnné, objekty nebo vestavěné funkce se podobají jazyku Java a jeho funkcionální model byl inspirován jazyky Scheme a Self. V Javascriptu jsou funkce k dispozici jako běžný datový typ a staví na prototypové dědičnosti. To znamená, že v Javascriptu se nevyužívají třídy, ale objekty, které mají vlastní atributy a metody. (Žára, 2015)

Již během několika prvních let, kdy se JavaScript objevil, předběhl v popularitě snad jediný programovací jazyk zaměřený na skriptování webových stránek, který mu mohl

[6] ECMA – z anglického European Computer Manufacturers Association. Evropská asociace výrobců počítačů.
[7] ES6 – z anglického ECMAScript6. Nejnovější aktualizace jazyka JavaScript.

konkurovat, VBScript[8] od společnosti Microsoft. Dnes je implementace Javascriptu na internetových prohlížečích nedílnou součástí. (Žára, 2015)

Z Javascriptu se v posledních letech vyvinul jeden z nejpopulárnějších programovacích jazyků v kontextu tvorby webových aplikací na světe. Lionel Sujay Vailshery (2021) na webové stránce statista uveřejnil graf využívání jazyků pro tvorbu webu, kde se JavaScript umístil na prvním místě. Téměř 65 procent z 83 tisíc respondentů odpovědělo, že k vytváření webu používají JavaScript. Dále se umístil Python s 48 procenty a SQL s 47 (viz Graf 1).



Graf 1: Graf používání programovacích jazyků pro tvorbu webových stránek

[8] VBScript – z anglického Visual Basic Script. Skriptovací jazyk pro tvorbu webových stránek.

JavaScript je všudypřítomný. Tím myslím, že se nachází snad na všech osobních počítačích po celém světe. Je to právě díky moderním internetovým prohlížečům, které jazyk JavaScript samy implementují, a tak JavaScript získává naprosto jedinečné postavení mezi programovacími jazyky. V posledních letech se mu také do určité míry podařilo nahradit zásuvný modul Flash. (Pehlivanian, 2014)

3 Javascriptové frameworky

JavaScript v podstatě za všechno vděčí příchodu XMLHttpRequestu[9] a schopnosti vytváření asynchronních volání na server bez nutnosti aktualizace stránky. Tato skutečnost umožnila naprosto novou uživatelskou zkušenost s používáním webových stránek, a právě zde se zrodily první Javascriptové frameworky, které zlepšily práci s Javascriptem takovým způsobem, aby se v něm daly vytvářet webové aplikace jednodušeji, a především aby rozšířili škálu jeho možností. Všechny moderní Javascriptové frameworky vznikly právě díky knihovně jQuery, která udělala velký kus práce a odstranila všechny rozdíly v prohlížečích a umožnila programátorům používat jen jednu zjednodušenou API dokumentaci na vytváření webových stránek. Za další bylo třeba webové stránky vytvořit tak, aby se chovaly a ovládaly, jako by to byly opravdové aplikace. Tento požadavek vytvořil spoustu dalších problémů a výzev. Knihovna jQuery sice poskytla spoustu nástrojů na ovládání DOM[10], ale už nenabízela žádný návod na to, jak si zorganizovat kód do struktury aplikace. Proto se často objevovaly jQuery aplikace, které měly přemíru nepřehledných souborů a kódů, které se téměř nedaly udržovat, číst, natož je dále rozšiřovat a zlepšovat. (Ruebbelke, 2015)

V tomto období začaly na povrch vyčnívat první Javascriptové frameworky, které měly pomoct s udržitelností obrovských bloků kódu. Tuto skutečnost nutně potřebovali především programátoři front-end[11] částí stránek. Front-end část stránky je část systému, kterou přímo ovládá uživatel, což jsou všechny zobrazující se na stránce. Dobře zpracovaná grafická část webové aplikace je potřeba především na sociálních sítích, jako Facebook, Instagram, Twitter apod. nebo na platformách poskytujících streamování filmů a seriálů, jako je HBO GO, Netflix nebo Prime Video.

V posledních několika letech začaly Javascriptové frameworky nabývat neuvěřitelné popularitě. JavaScript sice pohání web už přes 20 let, ale v dnešní době již pomalu přestává stačit na uživatelské i programátorské potřeby, a tak se spíše popularizují jeho frameworky. Ne však se všemi bylo jednoduché pracovat, a tak některé frameworky zanikly a upadly v zapomnění, ale byly zde samozřejmě i takové, kterým se podařilo prosadit a udržet na vrcholu díky jejich solidním možnostem pro tvorbu obrovských webových aplikací, jejich údržbu, následné rozšiřování a samozřejmě jednoduché testování. Mluvím zde zejména o frameworcích jako React, Angular, Express nebo Vue. (Ruebbelke, 2015)

---

[9] XMLHttpRequest – rozhraní starající se o komunikaci mezi serverem a klientem pomocí HTTP protokolu.
[10] DOM – z anglického Document Object Model. Objektově orientovaná reprezentace HTML dokumentu.
[11] Front-end – část systému, kterou ovládá uživatel.

V dnešní době jsou Javascriptové frameworky nedílnou součástí moderního front-end vývoje webových aplikací, jelikož poskytují programátorům již vyzkoušené a otestované nástroje pro tvorbu těchto aplikací, a tak ušetří spoustu času vymýšlením nástrojů vlastních. Frameworky také poskytují elementární základy pro tvorbu webových aplikací, a tak nemusí programátoři začínat od úplného začátku. Spousta společností pro tvorbu webových aplikací dnes frameworky standardně používá, a tím v podstatě nutí dnešní programátory se s nimi naučit pracovat. Často se objevuje názor, že frameworky nejsou zapotřebí, že obyčejný JavaScript dokáže úplně to stejné. Do určité části je to pravda, ale jak jsem již zmiňoval, frameworky programátorům zjednodušují práci obrovským způsobem. Zlepšení také nejde vidět jen na straně programátora, ale i na straně uživatele, protože frameworky zlepšují plynulost aktualizací prvků na stránce. Většina z těchto frameworků, které přežili konkurenční boje až dodnes, má v sobě také zabudované testovací servery, umožňují přetvořit různé části kódu do komponentů, které se jednodušeji volají a udržují, ale hlavně umožňují uživatelům přecházet z jedné stránky na druhou, aniž by bylo třeba načíst nový HTML soubor. (Understanding client-side JavaScript frameworks, 2022)

Pro srovnávání v této bakalářské práci jsem si vybral dva z nejpopulárnějších Javascriptových frameworků React a Angular. Vybral jsem si je především proto, že pro práci s nimi je zapotřebí naprosto odlišných znalostí a zkušeností, a proto mi připadají jako nejvhodnější kandidáti na srovnávání.

4 React

Obrázek 3: Logo frameworku React

V roce 2011 se vývojáři Facebooku střetli s několika problémy s údržbou jejich kódu. Jelikož se rozšiřovaly Facebookové reklamy, bylo pro stávající tým obtížné je udržovat, tak vedení spolčenosti muselo najmout další programátory, aby mohli jejich chod udržovat v co nejlepší kondici. Bohužel najmutí více programátorů měl opačný výsledek, než očekávali, a jelikož na kódu začalo pracovat více lidí s odlišnými zkušenostmi a návyky pro zápis kódu, začal být jejich kód nepřehledný, a to začalo zpomalovat společnost jako takovou. Po nějakém čase, kdy se vývojáři Facebooku již nemohli vyznat v jejich vlastním kódu, začali hledat nějaké řešení, které by jim pomohlo zefektivnit zápis a údržbu jejich kódu. V roce 2011 Jordan Walke představil základy Reactu. Již jako prototyp dokázal kód výrazně vylepšit, a to vedlo k jeho dalšímu rozšiřování. V roce 2012 si společnost Instagram všimla, že Facebook používá novou technologii, a tak požádali jejich vývojáře o sdílení Reactu s veřejností jako open-source software[12]. V roce 2013 tedy Jordan Walke představil a zpřístupnil React veřejnosti. (Hámori, 2022)

Dnes se zdá, že je React všude. Velké i malé společnosti jej používají pro tvorbu svých webových aplikací. Je takto populární hlavně proto, že React staví na obecných a základních dovednostech tvorby webových stránek. Neznamená to, že se v něm dá naučit během jednoho dne, nebo že jsou všechny jeho vlastnosti snadno pochopitelné na první pohled, místo toho React vyniká právě v tom, že minimalizuje množství potřebných vědomostí k tomu, aby v něm programátor mohl pracovat. Není potřeba se naučit žádné šablony, controllery ani komplexní vzorce, většina kódu je totiž psána za pomocí rozšíření jazyka JavaScript JSX[13], které využívá syntax v kombinaci s Javascriptem a standardním jazykem HTML. Takže při psaní kódu v Reactu si programátor utužuje mimo jiné také základy psaní webu. (Morgan, 2021)

React je populární Javascriptový framework pro tvorbu front-end webových aplikací. Dostává se mu takové popularitě, protože umožňuje vývojářům rychle vytvářet aplikace

[12] Open-source software – software, který je legálně dostupný široké veřejnosti.
[13] JSX – zkratka pro JavaScript XML. Rozšíření jazyka JavaScript o HTML značky.

za použití intuitivního stylu psaní kódu, který obsahuje JavaScript a syntax podobný jazyku HTML zvaný JSX. Dříve bylo složité začít nový projekt v Reactu. Bylo zapotřebí nastavit základní strukturu adresáře a transpiler (překladač), aby mohl převést moderní syntax Reactu do kódu, který by byl čitelný všemi prohlížeči. Dnes však existuje příkaz Create React App, který obsahuje všechny Javascriptové balíčky, které jsou zapotřebí ke zprovoznění React projektu. Příkaz také vytvoří testovací server, který se automaticky aktualizuje po úpravách kódu a umožňuje tak pohodlné testování aplikace. (Morgan, 2021)

4.1 React elementy a JSX

JSX je tedy jakási abstrakce jazyka HTML, která dovoluje programátorovi psát kód s podobnou syntaxí jako má jazyk HTML přímo v kódu Javascriptu a povoluje vytvářet React komponenty které vypadají jako standardní HTML značkování. JSX je vzorový jazyk elementů Reactu, a proto je základem pro jakékoliv značkování, které React vykreslí do vytvořené webové aplikace. Jelikož JSX umožňuje psát JavaScript kód do HTML značek, je výhodné využít jeho potenciál na maximum a používat v nich Javascriptové funkce a metody. (Morgan, 2021)

React elementy jsou nejmenší stavební bloky React aplikací. Element popisuje, co se má vykreslit na obrazovku. Na rozdíl od DOM elementů, React elementy jsou prosté objekty, a tak jsou jednoduché na vytvoření. React elementy jsou poměrně často zaměňovány za více známý koncept „komponenty“. Elementy jsou však menší části, a tak tyto komponenty tvoří. (Morgan, 2021)

4.2 Komponenty v Reactu

Komponenty jsou nezávislé kousky kódu plné řady funkcí, které se dají opakovaně využít ve vytvářené aplikaci, a dalo by se říct, že jsou to hlavní stavební kameny všech React aplikací, jelikož se bez nich programátor neobejde. Tyto komponenty mohou často být prosté Javascriptové funkce a třídy, ale spíše se využívají jako by to byly přizpůsobené HTML prvky. Tlačítka, nabídky menu a jakékoliv jiné front-end prvky mohou být všechny vytvořeny jako komponenty. Pokud se programátor naučí, jak vytvářet komponenty v Reactu, má už napůl vyhráno, protože pak bude schopen rozdělit komplexní aplikace do menších kousků, které jsou jednodušší na vytváření a údržbu. (Morgan, 2021)

4.2.1 Přizpůsobení komponentů pomocí argumentů props

Props jsou argumenty, které programátor zprostředkovává JSX elementům. Vypadají jako standardní HTML props, ale nejsou nijak předdefinované a mohou obsahovat mnoho různých JavaScriptových typů dat, jako jsou čísla, řetězce, funkce, nebo dokonce i jiné React komponenty. Vlastní komponenty mohou využívat props pro zobrazování dat nebo využít tato data k vytvoření interaktivity komponentů. Props jsou klíčovou částí k vytváření komponentů, které jsou adaptovatelné do různých situací. (Morgan, 2021)

4.2.2 Stylování komponentů

Stylování komponentů se dá provést hned několika způsoby. Buď vytvořením prostých kaskádových stylů užitím jazyka CSS, za pomoci JavaScriptových objektů pro stylování, nebo pomocí JSS, což je knihovna pro vytváření kaskádových stylů užíváním jazyka JavaScript. Každá z těchto možností má pochopitelně své výhody i nevýhody, některé zajišťují lepší protekci proti konfliktům ve stylech nebo umožňují přímé odkazování na props nebo jiná dynamická data. Všechny možnosti však mají jednu věc společnou: umožňují zanechání specifických stylů pro komponenty blízko těmto komponentům, což umožňuje jejich jednodušší opakované využívání po celém projektu nebo dokonce i přes několik na sobě nezávislých projektech. Všechny tyto možnosti spoléhají na vlastnosti jazyka CSS. Pro vytvoření stylů, které jsou integrovány s komponentem, se mohou aplikovat vložené objekty pro stylování, které používají jména CSS vlastností jako klíč a styl jako hodnotu. Nakonec je zde tedy možnost využití knihoven třetích stran jako je knihovna JSS, která umožňuje psát CSS kód v Javascriptové syntaxi. (Morgan, 2021)

4.2.3 Použití objektu state na třídy komponentů

State je v Reactu prostý Javascriptový objekt, který odkazuje na strukturu, která si udržuje přehled o tom, jak se v aplikaci v průběhu času mění data. Používání objektu state je zásadní dovednost v Reactu, jelikož umožňuje vytvářet interaktivní a dynamické webové aplikace. Jednoduše řečeno, objekt state má podobné vlastnosti a funkce jako argument props. Rozdíl je v tom, že state je privátní a kompletně ovládán komponentem. State se využívá pro všechno od sledování uživatelských vstupů až po zachytávání dynamických dat z API. (Morgan, 2021)

4.2.4 Správa komponentů pomocí souboru nástrojů Hooks

Při tvorbě aplikací pomocí Reactu, mít přehled o tom, jak se data v aplikaci mění postupem času se nazývá state management. Použití state objektů při vytváření aplikace zajistí, že aplikace bude dynamická a dokáže správně reagovat na vstupy zadávané uživatelem. Je zde spousta metod, jak používat state objekty při práci s Reactem, jako například využití knihovny třetí strany Redux, která pracuje na bázi tříd. Oficiální metoda Reactu pro zakomponování state objektů do komponentů se nazývá Hooks. Je to soubor nástrojů, který spouští vlastní funkce při změně argumentu props u komponentu. Jelikož pro používání této metody state managementu není potřeba využívat třídy, vývojáři tak mohou využít Hooks pro psaní kratších a lépe čitelných kódů, které jsou jednodušší pro sdílení s jinými vývojáři a lépe se udržují. Hlavní rozdíl mezi použití Hooks a Redux je takový, že při použití Hooks zde není žádný objekt, který by držel všechny state objekty. Místo toho se state objekty rozkládají do více menších kousků, které se mohou nezávisle na sebe aktualizovat. (Morgan, 2021)

4.2.5 Sdílení komponentů pomocí kontextu

State objekty se dají sdílet do více komponentů za použití kontextu. React kontext je rozhraní pro sdílení informací s ostatními komponenty, aniž by explicitně předával data jako props. To znamená, že informace mezi nadřazeným komponentem a hluboko vnořeným podřízeným komponentem se mohou mezi sebou sdílet, nebo ukládat data na jednom místě a dostat se k nim odkudkoli z aplikace. React kontext je dosti flexibilní na to, aby zvládl použít centralizovaný state management systém pro vytvářený projekt, nebo se dá rozdělit do menších sekcí aplikace. Díky kontextu mohou programátoři sdílet data po celé aplikaci bez používání jakýchkoli nástrojů třetích stran a s minimálním množstvím konfigurací. To poskytuje jednodušší alternativu oproti nástrojům jako již zmíněný Redux, který je dobrý pro vytváření větších a komplexnějších aplikací, ale zahrnuje také příliš mnoho nastavování a není proto nejvýhodnější jej používat pro malé nebo středně velké projekty. (Morgan, 2021)

4.3 Ovládání DOM a okénkových událostí

Při tvorbě webových stránek, eventy představují akce, které se dějí v internetovém prohlížeči. Odpovídáním na tyto eventy za pomoci event handlerů se dají vytvářet dynamické Javascriptové aplikace, které reagují na jakoukoli akci uživatele, včetně kliknutí nebo rolování myší na webové stránce, dotýkání se dotykového displeje a mnoho dalších akcí. V React

aplikacích se event handlery dají použít pro aktualizaci dat na state objektech nebo k zabránění defaultních akcí prohlížeče. K tomu React využívá tzv. SyntheticEvent obal místo normálního eventového prostředí. SyntheticEvent emuluje standardní eventy prohlížeče, ale poskytuje více konzistentní chování pro jiné internetové prohlížeče. React také poskytuje nástroje, které bezpečně přidají nebo odstraní okénkový event listener, když se komponent napojí nebo odpojí z DOM a tím dává programátorovi kontrolu nad okénkovými eventy, zatím co brání paměťovým únikům ze špatně odstraněných listenerů. (Morgan, 2021)

4.4 React forms

Forms jsou další zásadní součást tvorby webových aplikací s Reactem. Jelikož většina React aplikací jsou v zásadě tvořeny jako SPA[14], neodchází informace přímo z forms elementu na server. Místo toho se forms informace zachytávají na straně klienta a následně se posílají, nebo se zobrazují za pomoci přidaného Javascriptového kódu. React forms představují unikátní výzvu, protože se buď dají nastavit takovým způsobem, aby prohlížeč zpracovával většinu forms prvků a sbíral data skrze změny eventů v Reactu, nebo lze nastavit tak, aby React přímo a plně kontroloval elementy nastavením a aktualizací vložených hodnot. První přístup se nazývá nekontrolovaný komponent, protože React nenastavuje jeho hodnoty. Druhému přístupu se říká kontrolovaný komponent, protože React aktivně aktualizuje vstupní data uživatelů. (Morgan, 2021)

4.5 Použití metody Redux

Redux je v Javascriptu a Reactu populární metoda pro ukládání dat. Následuje hlavní zásadu, která říká, že svázání dat by mělo proudit v jednom směru a mělo by být uloženo jako jeden a jediný zdroj pravdy. Redux nabral na popularitě díky jednoduchému designu jeho konceptu a relativně malé implementaci. Redux operuje podle několika konceptů. Za prvé, úložný prostor je jen jedno místo s poli pro každou selekci dat. Data se aktualizují odesláním akce, která říká, jak se data budou měnit. Poté se vkládají akce a aktualizují se data za použití tzv. reducerů. Reducer je funkce, která aplikuje určité akce na data a vrací nový state objekt, místo toho, aby přetvářela ten starý. V malých aplikacích, kde není potřeba globálně ukládat data se doporučuje spíše použít mix lokálních state objektů a kontextu, který tyto state objekty bude řídit. (Morgan, 2021)

[14] SPA – z anglického Single Page Application. Jednostránková webová aplikace.

5 Angular

Obrázek 4: Logo frameworku Angular

V roce 2010, zaměstnanec společnosti Google Misko Hevery pracoval na vedlejším projektu. Tento projekt měl zjednodušit vytváření několika webových aplikací, na kterých pracoval. Svůj projekt ukázal svým přátelům programátorům, kteří mu ho pomohli upravit a vylepšit, a ke konci roku 2010 ho zpřístupnili veřejnosti jako Javascriptový framework Angular. Hned po jeho zveřejnění Angular nabyl na obrovské popularitě, a některé větší společnosti se ho chopily a začaly za jeho pomocí přetvářet své weby. V letech 2014 a 2015 Angular narazil na problém. Pro JavaScript vyšly nové standardy a vylepšení, se kterými si vývojáři Angularu nevěděli rady, jelikož tento framework byl opravdu vystavěn spíše jako vedlejší a malý projekt, a tak nebyl na takové aktualizace stavěný. Vývojáři z Googlu se ale nevzdali, a rozhodli se celý framework od základů předělat, aby se do budoucna vyhnuli podobným situacím. V roce 2016 se společnosti, které využívali Angular začaly obávat, že když Google vydá novou, od základů předělanou verzi Angularu, nebudou moci aktualizovat jejich stávající web na verzi novou. Proto se tyto společnosti obrátily spíše na konkurenční framework React. Po dlouhých čtyřech letech se v roce 2018 vývojářům povedlo vydat stabilní verzi Angularu. Čekání se ale vyplatilo, jelikož si s vývojem dali opravdu záležet, a od základů vybudovali framework, který poskytuje řešení pro všechny problémy, které by mohly při vytváření velké webové aplikace nastat. (Gavigan, 2018)

Dobrá organizace kódu, která umožňuje jeho údržbu, čitelnost, spolupráci s jinými programátory a další rozšiřování je v dnešní době obrovských webových aplikací velice žádané. Když je potřeba někam zařadit kód, který sbírá informace o tom, jak má vypadat uživatelské rozhraní, Angular má pro něj své místo. Je třeba automatizovat manipulaci s DOM? Angular má pro něj další vyhrazené místo. Framework Angular je postaven tak, aby měl každý blok kódu své intuitivní místo, kde by mohl existovat a dále se rozšiřovat, když je tomu třeba. Testování kódu je samozřejmostí. Angular byl od základů navržen tak, aby bylo možné jej testovat. Je jasné, že se chybám v kódu plně vyhnout nedá, ale pokud nějaký framework nabízí testování svého kódu, je určitě na místě toho využít. HTML jazyk je ve své podstatě dosti

limitovaný jazyk, který byl navržen k usnadnění rozvržení a struktuře ne moc komplexních interakcí. Jinými slovy, nebyl vytvořen, aby fungoval v dnešním světě moderních webových aplikací tak jak je známe. Některé frameworky se snaží obejít tato omezení úplným oproštěním se od jazyka HTML do řetězců nebo do preprocesorového dialektu. Problém je ale v tom, že HTML je ve skutečnosti výborný jazyk, co se deklaračního mechanismu týče. Angular proto jazyk HTML zahrnuje, a tak dává vývojářům možnost překonat jeho omezení a rozšířit jej do takové podoby, jaké je třeba. (Ruebbelke, 2015)

Angular je HTML rozšíření pro webové aplikace. To znamená, že i v nejzákladnější úrovni programování s frameworkem Angular se programátor setká s příhodným navazováním jazyka HTML s Javascriptovými objekty a jejich synchronizovaným udržováním. To také znamená, že Angular mimo jiné poskytuje vlastní HTML značky a elementy, známé také jako direktivy, které se dají použít pro velmi snadné a rychlé vytvoření dynamické webové aplikace. Vestavěné Angular direktivy pochopitelně nemohou být použity na všechny možné scénáře reálného světa, proto tým vývojářů frameworku Angular umožnil ostatním programátorům vytvářet své vlastní direktivy. Zde vztah s Angularem expanduje za hranice pouhého zprostředkovatele jednoduššího psaní kódu. Psaní kódu je potom mnohem rychlejší a intuitivnější, když si programátor může sám navrhnout HTML značky nebo atributy, kterými může vyjádřit cokoliv čeho se snaží dosáhnout. (Ruebbelke, 2015)

Angular je výborný framework pro vytváření jednostránkových webových aplikací. Tento typ aplikace se v posledních letech stává populárnějším a můžeme se tak s nimi setkat častěji. Za to je zodpovědný především vývoj HTML5 a dostupnost rychlejších internetových spojení. Díky jednostránkovým webovým aplikacím, mohou programátoři těchto aplikací dopřát responzivnější uživatelskou zkušenost s ovládáním webu a snížit nápor na webové servery. Je zde ale potencionální problém s webovými aplikacemi, které stahují velké množství obsahu z jednostránkového požadavku. Proto je zapotřebí tento obsah nějak zorganizovat a vést správným způsobem. Pomocí Angularu se tento problém dá vyřešit lehkým a elegantním způsobem, a to pomocí směrovacího systému. Ideálně by směrovací systém měl odhalit místa, kde se v aplikaci obsah nachází, a hlavně kdy tento obsah bude uživatel požadovat. Poté by měl systém obsah najít a načíst s minimálním množstvím kroků a bez zbytečných komplikací. (Grant, 2014)

5.1 Direktivy Angularu

Většina Javascriptových frameworků má své „speciální něco“, co je od sebe odlišuje. V případě Angularu jsou to bez pochyby direktivy. Direktivy jsou skvělým příkladem deklarativního programovacího stylu, který Angular tolik podporuje. Direktivy jsou klíčová část, jak pomocí tohoto frameworku dosáhnout skvělé uživatelské zkušenosti. Angular používá direktivy, aby rozšířil HTML o další funkcionality. Jednoduše řečeno, direktivy jsou příhodným způsobem, jak deklarativně volat Javascriptové funkce, a díky čistému vzhledu a intuitivní syntaxe jsou také velice oblíbené. Díky tomuto mocnému konceptu Angular také umožňuje vytvářet úplně nové direktivy, které se dají použít k zapouzdření logiky a zjednodušení manipulace s DOM. Jelikož dokážou přeměnit nebo dokonce vytvořit nové chování HTML, dají se využít pro cokoliv, ať už jde o jednoduché opětovné používání bloků statického obsahu nebo vytvoření sofistikovaného uživatelského rozhraní na straně klienta a jeho propojení s databází. Vestavěné direktivy poskytují obecnou úroveň funkcionalit a díky možnosti vytváření vlastních direktiv by se dalo říct, že jediným omezením je jen programátorova představivost. (Grant, 2014)

5.2 Výrazy Angularu

Výrazy Angularu jsou velice podobné těm Javascriptovým. Základem dobrých aplikací je schopnost zachytit a převést uživatelovy interakce vhodnými daty a kontextem. Angular má již pro tyto příležitosti některé vestavěné direktivy, jako ngClick, ngBlur, ngFocus, nebo ngSubmit. Angular také zachytává uživatelovy interakce vázáním vstupu formulářových elementů do datových struktur za použití direktivy ngModel. To umožňuje oboustranné vázání dat vstupu a vlastností controllerů, takže se změny budou šířit automaticky. (Ruebbelke, 2015)

5.3 Systém MVC[15]

Jedna z hlavních sil Angularu vězí v jeho schopnosti pomáhat programátorovi patřičně organizovat a strukturovat vytvářenou aplikaci. Malé aplikace z tohoto systému nebudou těžit tolik jako středně velké nebo velké aplikace, ale i tak by měly být správně strukturované. Jak již bylo naznačeno, systém MVC pomáhá jak správnému organizování aplikace,

[15] MVC – z anglického Model View Controller. Systém pro organizování a strukturaci zdrojového kódu.

tak i jejímu strukturování. Tento systém má tedy tři základní části: Model, Pohled a Controller. (Grant, 2014)

5.3.1 Model

Model reprezentuje základní logickou strukturu dat v softwarové aplikaci. Je poměrně častá chyba myslet na model jako na databázi za vytvářenou aplikací, spíše je lepší na něj pohlížet jako na model a část kódu která data reprezentuje. (Grant, 2014)

5.3.2 Pohled

Pohled je část kódu, která reprezentuje uživatelské rozhraní. Všechno, co uživatel vidí a vše na co může uživatel reagovat, jako tlačítka, dialogová okna a podobně. Aplikace obvykle mívají hned několik pohledů, a každý z nich reprezentuje určitou část modelu. (Grant, 2014)

5.3.3 Controller

Controller je Javascriptový objekt a jeho význam se dá odvodit již od jeho názvu. Kontroluje a ovládá data v aplikacích Angularu a dá se představit jako prostředník pohledu a modelu. (Grant, 2014)

Praktická část

6 Srovnání různých webových aplikací jednotlivých frameworků

Ještě před tím, než srovnám frameworky v kontextu své webové aplikace, ukážu za využití nástroje **Lighthouse** od společnosti Google srovnání snad nejznámějších a nejtypičtějších webových aplikací pro jednotlivé frameworky. Nejvíce charakteristické webové aplikace, které využívají framework React jsou **Facebook** a **Netflix** a z aplikací pro framework Angular jsou to webové aplikace **Gmail** a **PayPal**.

6.1 Kritéria nástroje Lighthouse

Nástroj Lighthouse používají vývojáři pro nalezení případných chyb a jejich opravení. V tomto případě se ale zaměřím jen na jeho statistiky čtyř kritérií, a to **načtení prvního prvku stránky, největšího prvku, čas pro načtení interaktivity a rychlostní index.**

Načtení prvního prvku stránky (First Contentful Paint) je kritérium měřící čas, který potřebuje prohlížeč pro načtení prvního prvku DOM poté, co se uživatel na stránku odkáže. Nástroj Lighthouse vyhodnotí skóre jako percentil na základě porovnání aktuálně zkoumané stránky a již testovaných stránek v databázi. Například stránky, které se pohybují v percentilu 99 načítají svůj první prvek do 1,2 sekundy a za dobré se považuje načtení do 1,8 sekundy (viz Tabulka 2). (First Contentful Paint, 2021)

FCP time (in seconds)	Color-coding
0–1.8	Green (fast)
1.8–3	Orange (moderate)
Over 3	Red (slow)

Tabulka 2: Tabulka hodnocení pro načtení prvního prvku stránky

Načtení největšího prvku stránky (Largest Contentful Paint) je další kritérium měřící čas, který potřebuje prohlížeč pro načtení prvku stránky. Nyní se však jedná o největší prvek stránky, může to být tabulka s textem nebo blok textu, ale nejčastěji se jedná o obrázky nebo videa. Skóre tohoto kritéria se opět hodnotí percentilem srovnáním s databází ostatních webových aplikací. Dobré skóre je považováno do percentilu 75, což odpovídá potřebné době pro načtení do 2,5 sekundy (viz Tabulka 3). (Walton, 2020)

LCP time (in seconds)	Color-coding
0-2.5	Green (fast)
2.5-4	Orange (moderate)
Over 4	Red (slow)

Tabulka 3: Tabulka hodnocení pro načtení největšího prvku stránky

Čas pro načtení interaktivity (Time to Interactive) je jedno z nejdůležitějších kritérií. Webové aplikace jsou dnes stavěné na tom, že jsou interaktivní a responzivní. Některé stránky se ale optimalizují hlavně vizuálně na úkor interaktivity. To může způsobit špatnou uživatelskou zkušenost, kdy se stránka jeví jako připravená pro uživatelské příkazy, ale když se s ní uživatel pokusí komunikovat, stránka na něj nereaguje. Kritérium tedy měří čas, kdy se stránka stane plně interaktivní, což znamená, že stránka ukazuje užitečný obsah, event handlery jsou zaregistrované pro většinu viditelných elementů a stránka reaguje na uživatelské příkazy do padesáti milisekund. Opět zde máme hodnocení v percentilu, kde při percentilu 99 se interaktivita načte za 2,2 sekundy a za dobré se považuje načtení až do 3,8 sekundy (viz Tabulka 4). (Time to Interactive, 2021)

TTI metric (in seconds)	Color-coding
0–3.8	Green (fast)
3.9–7.3	Orange (moderate)
Over 7.3	Red (slow)

Tabulka 4: Tabulka hodnocení pro načtení interaktivity

Poslední kritérium, které budu hodnotit je **rychlostní index** (Speed index), který odpovídá času potřebnému na vizuální zobrazení během načítání stránky. Nástroj Lighthouse jej měří tak, že zachytí video načítání stránky v prohlížeči a vypočítá vizuální vývoj mezi snímky a poté použije modul Speedline Node.js na vygenerování skóre rychlostního indexu. Načtení stránky až do 3,4 sekundy se ještě považuje za dobré (viz Tabulka 5). (Speed Index, 2021)

Speed Index (in seconds)	Color-coding
0–3.4	Green (fast)
3.4–5.8	Orange (moderate)
Over 5.8	Red (slow)

Tabulka 5: Tabulka hodnocení rychlostního indexu

6.2 Testování webových aplikací nástrojem Lighthouse

Webová aplikace **Facebook,** která byla nejpopulárnější sociální sítí před příchodem Tik Toku a Instagramu je kolébkou frameworku React, a tak není divu, že je jím také stále spravována. Měření nástrojem Lighthouse dopadlo takto (viz Tabulka 6): Načtení prvního prvku stránky zabralo 1,8 sekundy, což je považováno za dostatečné. Čas pro načtení interaktivity se také rovná 1,8 sekundy, ale v tomto případě to znamená výsledek dobrý.

Největší prvek stránky byl načten po dvou sekundách, což je dobrý výsledek a rychlostní index vyšel také dobře s časem 1,8 sekundy.

METRICS	Expand view
First Contentful Paint 1.8 s	Time to Interactive 1.8 s
Speed Index 1.8 s	Total Blocking Time 0 ms
Largest Contentful Paint 2.0 s	Cumulative Layout Shift 0

Tabulka 6: Tabulka načítání Facebooku

Webová aplikace **Netflix**, která je také spravována pomocí frameworku React, je jedna z nejoblíbenějších platforem pro streamování filmů a seriálů. Měření nástrojem Lighthouse dopadlo takto (viz Tabulka 7): Načtení prvního prvku stránky zabralo 0,7 sekundy, což je překvapivě dobrý výsledek, vzhledem k složitému vizuálnímu zpracování stránky. Čas pro načtení interaktivity se rovná 4,3 sekundy, což je dostačující. Největší prvek stránky byl načten po 2,3 sekundy, což je také dostačující výsledek a rychlostní index vyšel také dobře s časem 2,5 sekundy.

METRICS	Expand view
First Contentful Paint 0.7 s	Time to Interactive 4.3 s
Speed Index 2.5 s	Total Blocking Time 160 ms
Largest Contentful Paint 2.3 s	Cumulative Layout Shift 0.004

Tabulka 7: Tabulka načítání Netflixu

Webová aplikace **Gmail** od společnosti Google je e-mailová služba spravována pomocí frameworku Angular. Měření nástrojem Lighthouse dopadlo takto (viz Tabulka 8): Načtení prvního prvku stránky zabralo 0,4 sekundy, což je dobrý výsledek. Čas pro načtení interaktivity se rovná 5,4 sekundy, což je dostačující výsledek, ale překvapivě vysoký na to, že se jedná

pouze o e-mailovou službu. Největší prvek stránky má dobrý výsledek s načtením po 0,4 sekundy a rychlostní index vyšel také dobře s časem 2,3 sekundy.

METRICS	Expand view
● First Contentful Paint 0.4 s	▲ Time to Interactive 5.4 s
▲ Speed Index 2.3 s	▲ Total Blocking Time 430 ms
● Largest Contentful Paint 0.4 s	● Cumulative Layout Shift 0

Tabulka 8: Tabulka načítání Gmailu

Webová aplikace **PayPal** provozuje finanční služby v podobě internetového platebního systému a je spravována frameworkem Angular. Měření nástrojem Lighthouse dopadlo takto (viz Tabulka 9): Načtení prvního prvku stránky zabralo 0,8 sekundy, což je dobrý výsledek. Čas pro načtení interaktivity byl velice dobrý a krátký s časem 0,8 sekundy. Největší prvek stránky má dobrý výsledek s načtením po 0,8 sekundy a rychlostní index vyšel také velice dobře s časem 1,3 sekundy.

METRICS	Expand view
● First Contentful Paint 0.8 s	● Time to Interactive 0.8 s
● Speed Index 1.3 s	● Total Blocking Time 0 ms
● Largest Contentful Paint 0.8 s	● Cumulative Layout Shift 0

Tabulka 9: Tabulka načítání PayPalu

Z uvedených srovnání lze vidět, že výsledky jsou velice odlišné. Je to právě proto, že tyto webové aplikace jsou velice různorodé. Každá z uvedených aplikací se zaměřuje na jiné služby, jiný grafický vzhled stránek apod. Proto jsem následující webovou aplikaci v podobě kvízu na znalosti z anglického jazyka vytvořil identicky v obou frameworcích.

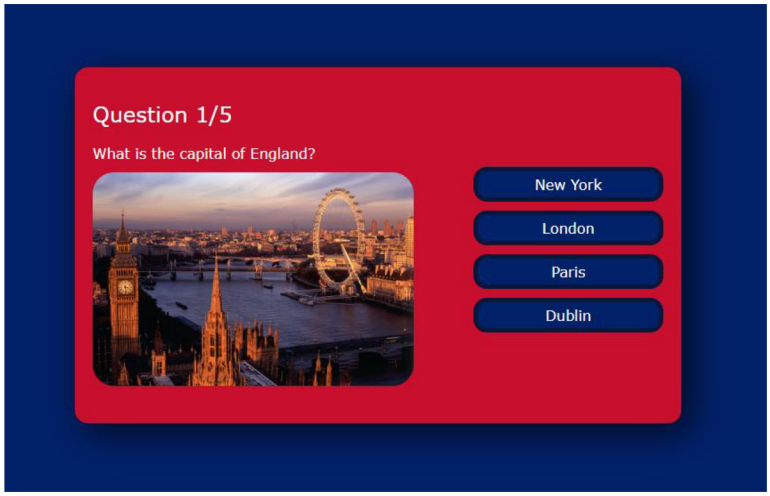
7 Jednoduchá webová aplikace

Odkazy na webové aplikace:

React: https://joyful-frangollo-6651e8.netlify.app/

Angular: https://beamish-madeleine-f329a3.netlify.app/

Jak jsem již zmiňoval, tuto webovou aplikaci jsem vyhotovil ve formě kvízu znalostí z anglického jazyka jako návaznost na můj druhý obor. Po načtení webové aplikace se již objeví první otázka (viz Obrázek 5). Uživatel má u každé otázky na výběr ze čtyř možností, kde je vždy jen jedna odpověď správná.



Obrázek 5: Screenshot první otázky kvízu

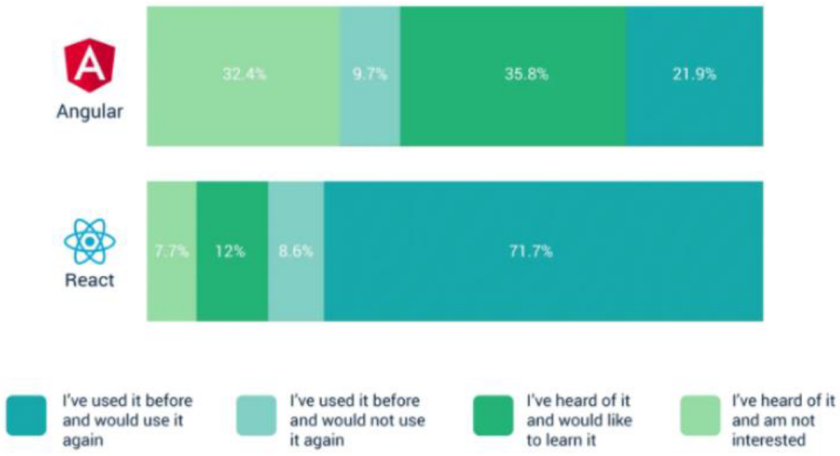
Po zodpovězení všech otázek se uživateli zobrazí stránka, která obsahuje tlačítko pro opětovné spuštění kvízu, uživatelovo dosažené skóre umístěné nahoře uprostřed, správné výsledky na levé straně a uživatelovy odpovědi na straně pravé. (viz Obrázek 6).

Obrázek 6: Screenshot konečné stránky s výsledky

Hodnocení jednotlivých kritérií budu hodnotit třemi stupni jako v předchozím srovnávání, a to jako dobré, dostačující a špatné. Opět budu k srovnávání používat open-source nástroj Lighthouse od společnosti Google na zachycení **rychlosti načtení prvního prvku stránky, nejobjemnějšího prvku stránky, rychlostní index a čas pro načtení interaktivity** (viz kapitola 6.1 Kritéria nástroje Lighthouse). **Dále budu srovnávat zatížení paměti** pomocí správce úloh v prohlížeči. **Jako poslední srovnám samotný zdrojový kód, složitost při vytváření nového projektu, následná orientace v projektu, přehlednost při psaní kódu, počet řádků potřebných pro naprogramování webové aplikace a kvalitu dostupné dokumentace.**

7.1 Stanovení předpokladů

Předpoklad č. 1: Předpokládám, že s frameworkem React se pracuje lépe než s frameworkem Angular. S předpokladem vycházím z příspěvku uživatele Suman (2021) na webu JavaScript in Plain English, kde uvádí graf popularity frameworků Angular a React (viz Graf 2). Respondentům bylo položeno několik otázek. V otázce, jestli programátoři již někdy použili daný framework a použili by jej znova odpovědělo u Reactu téměř 72 procent respondentů kladně, kdežto u Angularu to bylo jen necelých 22 procent.



Graf 2: Graf popularity jednotlivých frameworků

Předpoklad č. 2: Předpokládám, že framework React je celkově vhodnější volbou pro programování jednoduchých webových aplikací než Angular. Opět vycházím z příspěvku uživatele Suman (2021), který zmiňuje, že Angular je vhodný spíše pro tvorbu velkých

webových aplikací, kde může nabídnout bezpečnou a škálovatelnou platformu. U Reactu dále popisuje, že umožňuje programátorům vytvořit lehké/jednoduché aplikace v krátkém čase díky jeho intuitivní syntaxe.

7.2 Srovnání frameworků za pomoci nástroje Lighthouse

Webová aplikace, kterou jsem vytvořil za pomocí frameworku React v prováděném testu nástrojem Lighthouse dopadla následovně (viz Tabulka 10). Jelikož se jedná o poměrně jednoduchou webovou aplikaci, která v sobě nemá zakomponovaných mnoho složitých prvků, **všechna kritéria pro načítání dopadla dobře**. **První prvek se načetl po 0,3 sekundy, největší za 0,8 sekundy, rychlostní index za 0,3 sekundy a čas pro načtení interaktivity byl 0,3 sekundy.**

METRICS

First Contentful Paint 0.3 s	Time to Interactive 0.3 s
Speed Index 0.3 s	Total Blocking Time 0 ms
Largest Contentful Paint 0.8 s	Cumulative Layout Shift 0.045

Tabulka 10: Tabulka testování frameworku React nástrojem Lighthouse

Webová aplikace, kterou jsem vytvořil za pomocí frameworku Angular v prováděném testu nástrojem Lighthouse dopadla takto (viz Tabulka 11). Na první pohled je zde vidět, že časy jednotlivých kritérií se v obou frameworcích až tolik neliší, některé jsou dokonce stejné. **První prvek se načetl po 0,2 sekundy, největší za 0,7 sekundy, rychlostní index za 0,3 sekundy a čas pro načtení interaktivity byl 0,2 sekundy. Hodnotím tedy opět všechna kritéria jako dobrá**. Je tedy jasné, že zde žádný patrný rozdíl není, jelikož desetina sekundy je v tomto případě zanedbatelná. Je třeba se pro lepší srovnání podívat i na další kritéria.

METRICS

First Contentful Paint 0.2 s	Time to Interactive 0.2 s
Speed Index 0.3 s	Total Blocking Time 0 ms
Largest Contentful Paint 0.7 s	Cumulative Layout Shift 0.012

Tabulka 11: Tabulka testování frameworku Angular nástrojem Lighthouse

	React		Angular	
	Slovně	Číselně	Slovně	Číselně
Načtení prvního prvku	Dobré	3	Dobré	3
Načtení největšího prvku	Dobré	3	Dobré	3
Načtení interaktivity	Dobré	3	Dobré	3
Rychlostní index	Dobrý	3	Dobrý	3
Celkem	**12**		**12**	

Tabulka 12: Tabulka vyhodnocení aplikací nástrojem Lighthouse

7.3 Srovnání zatížení paměti jednotlivými frameworky

Při srovnání zatížení paměti byly výsledky obou frameworků rovněž dosti podobné (viz Obrázek 7). Ačkoliv aplikace ve frameworku Angular zabírá asi o 3 MB paměti více, **vyhodnotil jsem skóre u obou aplikací jako dobré**, jelikož v tomto případě je to zanedbatelný rozdíl. Samozřejmě při manipulaci se stránkou zde také hraje roli využití procesoru, tím pádem se navyšuje zatížení paměti, ale tyto údaje jsou rovněž velice podobné, proto je zde ani neuvádím.

Obrázek 7: Screenshot zatížení paměti oběma frameworky

	React		Angular	
	Slovně	Číselně	Slovně	Číselně
Zatížení paměti	Dobré	3	Dobré	3
Celkem	**3**		**3**	

Tabulka 13: Tabulka vyhodnocení zatížení paměti aplikací

7.4 Srovnání samotných projektů

Až do této chvíle jsem srovnávání soustředil spíše na uživatelskou část webové aplikace, kde naměřené výsledky byly téměř shodné. Byť je to velice důležité kritérium, které jsem vzal v potaz, ještě snad důležitější je webovou aplikaci správně naprogramovat. K tomu si musím vybrat framework, ve kterém se bude tato aplikace pohodlně, rychle a efektivně tvořit.

V této kapitole se tedy zaměřím na srovnání tvorby webové aplikace ve frameworcích React a Angular z pohledu programátora. Aplikaci jsem vytvářel v bezplatném editoru zdrojového kódu Visual Studio Code a pro spouštění a testování programu jsem využil softwarový systém Node.js.

Jelikož se JavaScript na téměř žádných školách nevyučuje, nedivím se, že jeho frameworky jsou pro většinu lidí ještě více neznámé. Proto jsem, stejně jako většina programátorů pro naučení potřebné syntaxe, metod a funkcí využil oficiální dokumentace frameworků. Když jsem v oficiální dokumentaci řešení určitého problému nemohl dohledat, využil jsem programátorské komunity jako Stack Overflow nebo výukové stránky jako W3Schools.

7.4.1 Vytvoření nového projektu

Jako první věc jsem musel vytvořit nový projekt. K tomu mi posloužil návod v dokumentaci frameworků. V obou dokumentacích je návod na vytvoření projektu jasný a srozumitelný (viz Obrázky 8 a 9), tudíž **vytvoření nového projektu hodnotím v obou frameworcích jako dobré**.

Create React App

Create React App is a comfortable environment for **learning React**, and is the best way to start building **a new single-page application** in React.

It sets up your development environment so that you can use the latest JavaScript features, provides a nice developer experience, and optimizes your app for production. You'll need to have Node >= 14.0.0 and npm >= 5.6 on your machine. To create a project, run:

```
npx create-react-app my-app
cd my-app
npm start
```

Obrázek 8: Screenshot kódu pro vytvoření nového React projektu

Create a new workspace and an initial application

You develop applications in the context of an Angular workspace. A workspace contains the files for one or more projects. A project is the set of files that comprise an application or a library. For this tutorial, you will create a new workspace.

To create a new workspace and an initial application project:

1. Ensure that you are not already in an Angular workspace folder. For example, if you have previously created the Getting Started workspace, change to the parent of that folder.
2. Run the CLI command `ng new` and provide the name `angular-tour-of-heroes`, as shown here:

```
ng new angular-tour-of-heroes
```

3. The `ng new` command prompts you for information about features to include in the initial application project. Accept the defaults by pressing the Enter or Return key.

The Angular CLI installs the necessary Angular `npm` packages and other dependencies. This can take a few minutes.

It also creates the following workspace and starter project files:

- A new workspace, with a root folder named `angular-tour-of-heroes`.
- An initial skeleton app project in the `src/app` subfolder.
- Related configuration files.

The initial app project contains a simple Welcome application, ready to run.

Serve the application

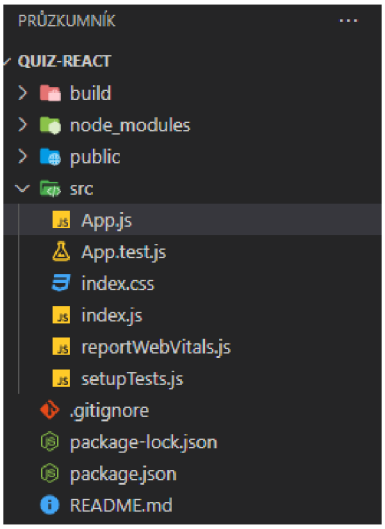
Go to the workspace directory and launch the application.

```
cd angular-tour-of-heroes
ng serve --open
```

Obrázek 9: Screenshot kódu pro vytvoření nového Angular projektu

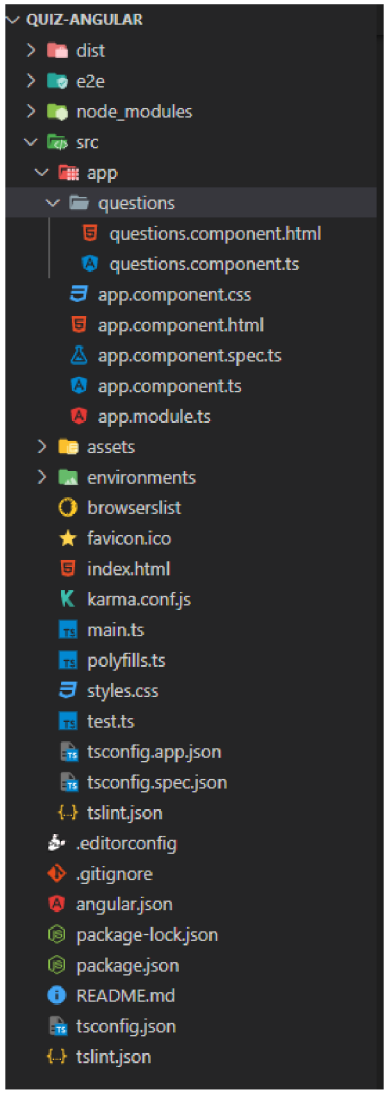
7.4.2 Orientace v projektu

Dalším krokem bylo spuštění projektu v editačním prostředí. V obou případech se jednalo o složku obalenou dalšími složkami a soubory, které se odkazují na online dokumentaci a řeší všechny nastavení a konfigurace pro propojení s internetovými prohlížeči. Jak lze vidět na Obrázku 10, **React projekt má složky logicky pojmenované a vytřízené a mimo ně má minimum souborů potřebných pro úpravy kódu aplikace, které jsou pojmenované tak, aby orientace v nich byla co nejjednodušší. Orientaci v React projektu hodnotím jako dobrou.**



Obrázek 10: Screenshot rozložení souborů v React projektu

Angular projekt na rozdíl od Reactu obsahuje příliš mnoho nesystematických složek a souborů (viz Obrázek 11), **které mohou na první pohled mást.** Této skutečnosti přispívá fakt, že Angular skutečně potřebuje mnohem více konfigurací, aby mohl v internetových prohlížečích pracovat správně, protože se zde pro psaní kódu využívá TypeScript, což je jakási nadstavba nad jazykem JavaScript, kterou internetové prohlížeče samy přeložit nedokážou. Je zde tedy zapotřebí mnohem více konfigurování kódu než v případě Reactu, a tak je zde spousta konfiguračních souborů navíc. **Orientaci v Angular projektu hodnotím spíše jako špatnou.**



Obrázek 11: Screenshot rozložení souborů v Angular projektu

7.4.3 Přehlednost při psaní kódu

V React projektu jsou všechny soubory pro tvorbu aplikace v jedné složce „src“ a na tyto soubory se dá mezi sebou jednoduše odkazovat. O renderování celé aplikace se stará soubor „index.js“, který se pomocí vlastní značky „<App />“ odkazuje na komponent v podobě funkce „App“, který je exportován ze souboru „App.js“. Nejlepším způsobem, jak vytvářet nejen webové aplikace, je rozdělení kódu do systému MVC, neboli tak, aby každý soubor zajišťoval určitou funkci v celkovém programu. Framework React tento systém přímo nevyžaduje, což je při tvorbě jednoduchých aplikací dobré, jelikož jsem nemusel řešit navíc žádné odkazování mezi soubory a **menší kód se dobře udržoval i v rozmezí jednoho souboru.**

Hlavním souborem této aplikace je tedy soubor „App.js“. V něm je vložen list, který obsahuje všechny otázky kvízu. Tyto otázky jsem musel nějakým způsobem zobrazit uživateli ve správném pořadí jednu po druhé. K tomu React využívá JSX, což je intuitivní rozšíření jazyku JavaScript, jehož syntax připomíná HTML kód. JSX je opravdu intuitivní, jelikož přímo do HTML kódu lze do složených závorek jednoduše zapsat Javascriptové funkce, metody, listy, proměnné apod. **Přehlednost při psaní kódu zde tedy hodnotím jako dobrou.**

V Angularu je rozložení souborů poněkud složitější. Také jsou zde, stejně jako v Reactu **všechny soubory pro tvorbu aplikace uloženy v jedné složce, a to ve složce „app“.** V této složce je několik souborů, které jsem spolu musel navzájem propojit. **Framework Angular nejenže podporuje tvorbu aplikací systémem MVC, ale dokonce jej přímo vyžaduje**. Bylo zde tedy nutné vytvořit soubory zvlášť, které slouží jako komponenty, a nakonec jsem je všechny navzájem propojil. Při tvorbě složitých a komplexních webových aplikací, které mají spoustu funkcí a velké množství kódu je tento systém výhodný z hlediska přehlednosti a udržitelnosti, nýbrž v tomto případě, kdy je aplikace myšlena jako malá, jednoduchá a kdy se nepředpokládá velké množství aktualizací je **tento systém zbytečně složitý a zdlouhavý.**

Bylo zde tedy třeba deklarovat komponenty do samostatného souboru „app.module.ts“ a až posléze je využívat dále. Komponenty obsahují zdrojový kód v TypeScriptu a jsou přímo propojeny se soubory HTML. Kód pro aktualizaci prvků webové stránky nejde v Angularu vkládat do HTML značek tak jednoduše, jako je tomu u Reactu. Sice jsem zde měl rovněž možnost kód zapsat do dvou složených závorek, ale tato možnost je výrazně omezena, jelikož lze do složených závorek zapisovat pouze příkazy vracející textový řetězec, a tak jsem musel předdefinovat všechny funkce a metody v komponentu, a až poté se na ně odkazovat. **Musel**

jsem zde tedy tvořen jakýsi logický řetězec odkazů, který je pro jednoduchou aplikaci poměrně zbytečný, a jelikož se tento systém nedá obejít, hodnotím přehlednost při psaní kódu pro tento typ aplikací jako dostačující.

7.4.4 Potřebný počet řádků zdrojového kódu

Pro jednoduchou orientaci v kódu jsem je psal takovým způsobem, aby se co nejvíce logiky vešlo do co nejméně řádků. Samozřejmě jsem počet řádků nemohl snižovat na úkor funkcí aplikace, jinak by to bylo spíše kontraproduktivní. Oba uvedené frameworky fungují na podobném principu. Základ tvoří HTML kód, který se načítá na obrazovku uživateli. **V případě Angularu jsem HTML kód vložil do samostatného HTML souboru o asi padesáti řádcích** „questions.component.html“. **Do tohoto základu jsem poté vložil funkce ze souboru o sto řádcích** „questions.component.ts“, aby se statická webová stránka stala dynamickou. **Aktualizace proměnných v Angularu funguje stejně jako u prostého Javascriptu, což je výhoda oproti frameworku React, kde jsem aktualizace prvků musel provádět pomocí objektů state.** Počet řádků není tedy nijak velký, a tak bych kód zvládnul upravovat i v rozmezí jednoho souboru, kdyby framework Angular přímo nevyžadoval rozdělení kódu do samotných souborů. **Nicméně celkovou velikost zdrojového kódu hodnotím dobře.**

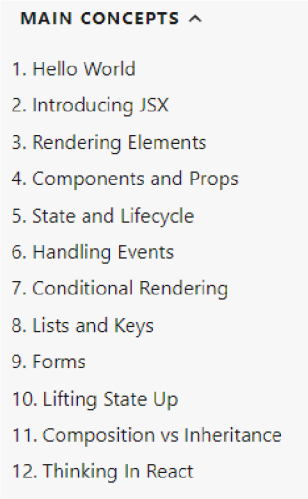
Zdrojový kód k webové aplikaci ve frameworku React jsem uložil do jediného souboru „App.js“. **Soubor obsahuje asi 150 řádků, což je srovnatelné s počtem řádků v Angularu.** V kódu je opět obsažen základ jazyka HTML, do nějž jsem vepsal funkce pro zajištění interaktivity. Byť je zdrojový kód jen v jednom souboru, musím podotknout, že práce s ním je zde poněkud obtížnější**. React vyžaduje přiřazování objektů state proměnným, které na těchto proměnných provádějí aktualizace za chodu programu.** State objekty zápis kódu sice obecně zjednodušují a zkracují, ale musel jsem dohledávat, jak je správně použít, což mi zabralo více času, a proto bych React nedoporučil jako framework pro jednorázové použití. Když jsem se ale s objekty state seznámil lépe, věřím, že bych další aplikaci zvládl naprogramovat mnohem rychleji. **Velikost zdrojového kódu ale hodnotím dobře.**

7.4.5 Kvalita dokumentace

Kvalitní dokumentace je v případě Javascriptových frameworků velice důležitá, jelikož se na školách povětšinou nevyučují. Kde lépe jsem měl tedy začít hledat informace o jejich funkcích, syntaxe a způsobu využití než na oficiálních stránkách. Především jsem

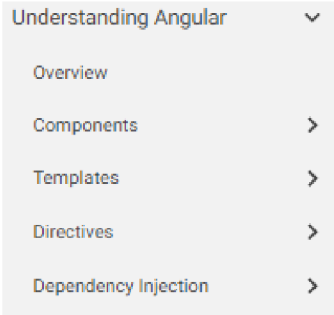
vyhledával technickou část dokumentace, což jsou dokumentace kódu, algoritmy a popis rozhraní.

V technické dokumentaci React se nachází několik kapitol (viz Obrázek 12). **Kapitoly jsou logicky rozděleny od nejjednodušších příkazů jako klasické vypsání do konzole „Hello World“ a zapisování elementů, přes vysvětlení React komponentů, až po kompozice**. Nejenže je dokumentace logicky segmentovaná, ale také tyto jednotlivé kapitoly obsahují jednoduché a srozumitelné vysvětlení všech potřebných pojmů, doprovázené o vzorové příklady právě představovaného konceptu. Navíc je zde také umístěn velice podrobný tutoriál k vytvoření první funkční webové aplikace v tomto frameworku. **Dokumentaci hodnotím jako dobrou.**



Obrázek 12: Screenshot technické dokumentace frameworku React

Technická dokumentace pro framework Angular vypadá dosti podobně (viz Obrázek 13). Nachází se na oficiálních stránkách frameworku, jsou zde kapitoly od nejjednoduššího konceptu po nejsložitější. Jednotlivé kapitoly v sobě obsahují podkapitoly, které jsem využil pro inspiraci při psaní mé webové aplikace. Také jsem zde našel návod na vytvoření první webové aplikace v podobě tutoriálu, který je popsaný krok za krokem. **Dokumentace je v případě Angularu také dobře zpracována.**



Obrázek 13: Screenshot technické dokumentace frameworku Angular

	React		Angular	
	Slovně	Číselně	Slovně	Číselně
Vytvoření nového projektu	Dobré	3	Dobré	3
Orientace v projektu	Dobrá	3	Špatná	1
Přehlednost při psaní kódu	Dobrá	3	Dostačující	2
Potřebný počet řádků zdrojového kódu	Dobrý	3	Dobrý	3
Kvalita dokumentace	Dobrá	3	Dobrá	3
Celkem	**15**		**12**	

Tabulka 14: Tabulka vyhodnocení programátorské části

7.5 Vyhodnocení srovnávání

Předpoklad č. 1: Předpokládám, že s frameworkem React se pracuje lépe než s frameworkem Angular.

Předpoklad č. 2: Předpokládám, že framework React je celkově vhodnější volbou pro programování jednoduchých webových aplikací než Angular.

Srovnání JavaScriptových frameworků React a Angular jsem provedl v několika fázích. Oba frameworky obstály stejně dobře ve všech kritériích při testování nástrojem Lighthouse i při měření zatížení paměti. Při srovnání práce s jednotlivými projekty se však v některých oblastech objevily výrazné rozdíly ve prospěch frameworku React (viz Tabulka 15).

	React	Angular
Vyhodnocení nástrojem Lighthouse	12	12
Vyhodnocení zatížení paměti	3	3
Vyhodnocení z programátorské části	15	12
Celkem	**30**	**27**

Tabulka 15: Tabulka celkového vyhodnocení jednotlivých frameworků

Z výše uvedené tabulky vyplývá, že ze strany uživatele výrazné rozdíly mezi jednotlivými frameworky nejsou. Z programátorského hlediska zde však lze vidět, že framework React dosáhl lepších výsledků než Angular, a tak přijímám oba stanovené předpoklady.

Závěr

V dnešní době již jazyky pro tvorbu webových stránek HTML a CSS jednoduše nestačí na uživatelské požadavky. Webovým stránkám je třeba zajistit dynamičnost a responzivitu, kterou umožňuje programovací jazyk JavaScript. V bakalářské práci jsem se na tento moderní přístup k vytváření webových stránek zaměřil, zejména tedy na použití Javascriptových frameworků.

V teoretické části bakalářské práce jsem věnoval pozornost historii tvorby webových stránek, Javascriptu jako takovému a jeho frameworkům. V kapitolách o jednotlivých frameworcích jsem se věnoval zejména bližšímu pohledu na způsob, jakým se v nich programuje. Šlo především o obecné seznámení se s nimi a pochopení jejich funkcí ve světě programování front-end částí webových aplikací. Při zpracovávání těchto kapitol jsem se jako nováček v tomto odvětví dozvěděl, že Javascriptové frameworky jsou určeny především pro práci s většími projekty, jelikož je zde nutné strukturovat zdrojový kód do mnoha souborů a složek, a přesně to tyto frameworky nabízejí. Frameworky používají internetové společnosti, které chtějí zaujmout grafickým vzhledem uživatelského rozhraní jejich webových aplikací, jejich responzivitou a dynamičností. Jedná se tedy o webové aplikace s širokou škálou grafického obsahu, především sociální sítě, aplikace poskytující streamovací služby apod. Důležité je to pro ně z finančního hlediska, jelikož většina těchto aplikací má vysokou část příjmů z reklam nebo v podobě předplatného, a to obecně znamená, že čím více uživatelů si tyto internetové společnosti získají, a především udrží po delší dobu, tím více peněz jim tyto složky budou vynášet. Na internetu potom nastává konkurenční boj společností s podobným obsahem, a to vede k vývoji dalších lepších frameworků a rozšiřování funkcionalit těch starých.

V praktické části práce jsem analyzoval data dvou frameworků React a Angular, které patří mezi nejpoužívanější a nejrozšířenější. Data jsem získával za pomocí využití nástrojů Lighthouse pro měření rychlosti načítání webových aplikací a správce úloh v prostředí internetového prohlížeče pro měření zatížení paměti. Dále jsem výzkum směřoval na srovnání zdrojového kódu, abych celkové srovnání uchopil s obou stran, jak ze strany uživatele, tak i programátora webové aplikace. Hlavním cílem mého výzkumu tedy bylo srovnat oba Javascriptové frameworky z pohledu uživatelské a programátorské části, a vyhodnotit, který z nich se více hodí pro naprogramování jednoduché webové aplikace. Po analyzování získaných dat jsem zjistil, že pro tento typ webové aplikace je vhodnější framework React. Rozdíly z pohledu uživatele byly minimální až žádné, proto hrálo klíčovou roli srovnání

zdrojového kódu. Aplikace byla opravdu jednoduchá, určitě by se na ní dalo navázat rozšířením o menu, ve kterém by bylo možné vybrat téma, například historie, osobnosti, zeměpis, literatura apod., navýšit počet otázek, volné místo zaplnit užitečným obsahem atd. Při tomto přístupu by ale nejspíš bylo nutné vytvořit také databázi a zakomponovat zde více nástrojů pro tvorbu webových aplikací, na což mé dosavadní zkušenosti s programováním nestačí, a tak na tento problém mohu navázat ve své diplomové práci.

Stanovil jsem si dva předpoklady k praktické části bakalářské práce, které jsem po srovnání všech kritérií jednotlivých frameworků potvrdil. V kapitole pro stanovení předpokladů sice uvádím jen jeden zdroj, ze kterého jsem vycházel, ale předpoklady jsem odvozoval také ze svých obecných znalostí a mého pozorování z několika na sobě nezávislých zdrojů při kompletaci této bakalářské práce.

Na závěr je nutno zmínit, že je v dnešní době nedostatek programátorů, kteří jsou ochotni se do Javascriptových frameworků ponořit více do hloubky. Je to způsobeno hlavně tím, že tomuto tématu není věnována dostatečná pozornost ani ze strany školství, ani ze strany veřejnosti, i přes skutečnost, že se stále více rozšiřují a využívají při tvorbě moderních webových aplikací a pomalu se tak stávají součástí každodenního života.

Seznam použitých zdrojů

BERNERS-LEE, Tim. Information Management: A Proposal. *W3* [online]. CERN, 1990 [cit. 2022-04-08]. Dostupné z: https://www.w3.org/History/1989/proposal.html

First Contentful Paint, 2021. *Web.dev* [online]. [cit. 2022-04-09]. Dostupné z: https://web.dev/first-contentful-paint/?utm_source=lighthouse&utm_medium=devtools

GAVIGAN, Dave, 2018. The History of Angular. In: *The Startup Lab* [online]. [cit. 2022-04-08]. Dostupné z: https://medium.com/the-startup-lab-blog/the-history-of-angular-3e36f7e828c7

GRANT, Andrew. *Beginning AngularJS*. New York: Apress Media, 2014. ISBN 9781484201619.

HÁMORI, Ferenc, 2022. The History of React.js on a Timeline. In: *RisingStack* [online]. [cit. 2022-04-08]. Dostupné z: https://blog.risingstack.com/the-history-of-react-js-on-a-timeline/

Informace o webových aplikacích, 2021. *Adobe* [online]. 2021 [cit. 2022-04-15]. Dostupné z: https://helpx.adobe.com/cz/dreamweaver/using/web-applications.html

JANOVSKÝ, Dušan. *Jak psát web* [online]. 1999–2022 [cit. 11. 3. 2022]. Dostupné z https://www.jakpsatweb.cz/css/

JavaScript History, 1999–2022. *W3Schools* [online]. [cit. 2022-04-09]. Dostupné z: https://www.w3schools.com/js/js_history.asp

LAURENČÍK, Marek. *Tvorba www stránek v HTML a CSS*. Praha: Grada, 2019. ISBN 9788027122417.

MORGAN, Joe. *How to code in React.js*. New York: DigitalOcean, 2021. ISBN 9781735831749.

PEHLIVANIAN, Ara a Don NGUYEN. *JavaScript Okamžitě*. Brno: Computer Press, 2014. ISBN 978802514163.

RUEBBELKE, Lukas. *AngularJS in Action*. New York: Manning Publications, 2015. ISBN 9781617291333.

Speed Index, 2021. *Web.dev* [online]. [cit. 2022-04-09]. Dostupné z: https://web.dev/speed-index/?utm_source=lighthouse&utm_medium=devtools

SUMAN, 2021. Angular vs. React: Who is Winning? In: *JavaScript in Plain English* [online]. 2021 [cit. 2022-04-18]. Dostupné z: https://javascript.plainenglish.io/angular-vs-react-who-is-winning-e9b482879e6a

Time to Interactive, 2021. *Web.dev* [online]. [cit. 2022-04-09]. Dostupné z: https://web.dev/interactive/?utm_source=lighthouse&utm_medium=devtools

Understanding client-side JavaScript frameworks, 2022. *Mdn* [online]. [cit. 2022-04-08]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks

WALTON, Philip, 2020. Largest Contentful Paint (LCP). In: *Web.dev* [online]. [cit. 2022-04-09]. Dostupné z: https://web.dev/lcp/#what-is-a-good-lcp-score

ŽÁRA, Ondřej. *JavaScript Programátorské techniky a webové technologie*. Brno: Computer Press, 2015. ISBN 9788025145739.

Seznam obrázků

Obrázek 1: Logo HTML a CSS 10
Obrázek 2: Logo Javascriptu 14
Obrázek 3: Logo frameworku React 19
Obrázek 4: Logo frameworku Angular 24
Obrázek 5: Screenshot první otázky kvízu 34
Obrázek 6: Screenshot konečné stránky s výsledky 34
Obrázek 7: Screenshot zatížení paměti oběma frameworky 38
Obrázek 8: Screenshot kódu pro vytvoření nového React projektu 39
Obrázek 9: Screenshot kódu pro vytvoření nového Angular projektu 40
Obrázek 10: Screenshot rozložení souborů v React projektu 41
Obrázek 11: Screenshot rozložení souborů v Angular projektu 42
Obrázek 12: Screenshot technické dokumentace frameworku React 45
Obrázek 13: Screenshot technické dokumentace frameworku Angular 46

Obrázek 1 dostupný z: https://www.pngitem.com/middle/iJJbhR_html5-and-css3-transparent-background-html-logo-hd/

Obrázek 2 dostupný z: https://upload.wikimedia.org/wikipedia/commons/9/99/Unofficial_JavaScript_logo_2.svg

Obrázek 3 dostupný z: https://upload.wikimedia.org/wikipedia/commons/4/47/React.svg

Obrázek 4 dostupný z: https://upload.wikimedia.org/wikipedia/commons/c/cf/Angular_full_color_logo.svg

Obrázky 5-7: vlastní zpracování

Obrázek 8 dostupný z: https://reactjs.org/docs/getting-started.html

Obrázek 9 dostupný z: https://angular.io/docs

Obrázky 10-11: vlastní zpracování

Obrázek 12 dostupný z: https://reactjs.org/docs/create-a-new-react-app.html

Obrázek 13 dostupný z: https://angular.io/tutorial/toh-pt0

Seznam Grafů

Graf 1: Graf používání programovacích jazyků pro tvorbu webových stránek 15
Graf 2: Graf popularity jednotlivých frameworků 35

Graf 1 dostupný z: https://www.statista.com/statistics/793628/worldwide-developer-survey-most-used-languages/

Graf 2 dostupný z: https://javascript.plainenglish.io/angular-vs-react-who-is-winning-e9b482879e6a

Seznam tabulek

Tabulka 1: Tabulka srovnání vlastností statického a dynamického webu 12
Tabulka 2: Tabulka hodnocení pro načtení prvního prvku stránky 29
Tabulka 3: Tabulka hodnocení pro načtení největšího prvku stránky 30
Tabulka 4: Tabulka hodnocení pro načtení interaktivity 31
Tabulka 5: Tabulka hodnocení rychlostního indexu 31
Tabulka 6: Tabulka načítání Facebooku 32
Tabulka 7: Tabulka načítání Netflixu 32
Tabulka 8: Tabulka načítání Gmailu 33
Tabulka 9: Tabulka načítání PayPalu 33
Tabulka 10: Tabulka testování frameworku React nástrojem Lighthouse 36
Tabulka 11: Tabulka testování frameworku Angular nástrojem Lighthouse 37
Tabulka 12: Tabulka vyhodnocení aplikací nástrojem Lighthouse 37
Tabulka 13: Tabulka vyhodnocení zatížení paměti aplikací 38
Tabulka 14: Tabulka vyhodnocení programátorské části 46
Tabulka 15: Tabulka celkového vyhodnocení jednotlivých frameworků 47

Tabulka 1 dostupná z: http://www.differencebetween.net/technology/difference-between-static-website-and-dynamic-website/

Tabulka 2 dostupná z: https://web.dev/first-contentful-paint/?utm_source=lighthouse&utm_medium=devtools

Tabulka 3 dostupná z: https://web.dev/lighthouse-largest-contentful-paint/?utm_source=lighthouse&utm_medium=devtools

Tabulka 4 dostupná z: https://web.dev/interactive/?utm_source=lighthouse&utm_medium=devtools

Tabulka 5 dostupná z: https://web.dev/speed-index/?utm_source=lighthouse&utm_medium=devtools

Tabulky 6-15: vlastní zpracování

Seznam příloh

Příloha 1: Slovník zkratek a pojmů

Příloha 1: Slovník zkratek a pojmů

1. HTML – z anglického HyperText Markup Language. Značkovací jazyk pro tvorbu webových stránek.
2. CSS – z anglického Cascading Style Sheets. Jazyk pro popis vizuálního zpracování webových stránek.
3. SQL – z anglického Structured Query Language. Jazyk pro tvorbu a úpravu databází.
4. PHP – z anglického Hypertext Prepocessor. Skriptovací programovací jazyk pro tvorbu webových stránek.
5. API – z anglického Application Programming Interface. Rozhraní pro programování aplikací.
6. ECMA – z anglického European Computer Manufacturers Association. Evropská asociace výrobců počítačů.
7. ES6 – z anglického ECMAScript6. Nejnovější aktualizace jazyka JavaScript.
8. VBScript – z anglického Visual Basic Script. Skriptovací jazyk pro tvorbu webových stránek.
9. XMLHttpRequest – rozhraní starající se o komunikaci mezi serverem a klientem pomocí HTTP protokolu.
10. DOM – z anglického Document Object Model. Objektově orientovaná reprezentace HTML dokumentu.
11. Front-end – část systému, kterou ovládá uživatel.
12. Open-source software – software, který je legálně dostupný.
13. JSX – zkratka pro JavaScript XML. Rozšíření jazyka JavaScript o HTML značky.
14. SPA – z anglického Single Page Application. Jednostránková webová aplikace.
15. MVC – z anglického Model View Controller. Systém pro organizování a strukturaci zdrojového kódu.